JupyterLab in Retrograde: Contextual Notifications That Highlight Fairness and Bias Issues for Data Scientists

Galen Harrison*
gh7vp@virginia.edu
University of Virginia
Charlottesville, VA, USA

Kevin Bryson
kbryson@uchicago.edu
University of Chicago
Chicago, IL, USA

Ahmad Emmanuel Balla Bamba
aebbamba@uchicago.edu
University of Chicago
Chicago, IL, USA

Luca Dovichi
lucadovichi@uchicago.edu
University of Chicago
Chicago, IL, USA

Aleksander Herrmann Binion
biniona@uchicago.edu
University of Chicago
Chicago, IL, USA

Arthur Borem
arthurborem@uchicago.edu
University of Chicago
Chicago, IL, USA

Blase Ur
blase@uchicago.edu
University of Chicago
Chicago, IL, USA

ABSTRACT

Current algorithmic fairness tools focus on auditing completed models, neglecting the potential downstream impacts of iterative decisions about cleaning data and training machine learning models. In response, we developed Retrograde, a JupyterLab environment extension for Python that generates real-time, contextual notifications for data scientists about decisions they are making regarding protected classes, proxy variables, missing data, and demographic differences in model performance. Our novel framework uses automated code analysis to trace data provenance in JupyterLab, enabling these notifications. In a between-subjects online experiment, 51 data scientists constructed loan-decision models with Retrograde providing notifications continuously throughout the process, only at the end, or never. Retrograde's notifications successfully nudged participants to account for missing data, avoid using protected classes as predictors, minimize demographic differences in model performance, and exhibit healthy skepticism about their models.

CCS CONCEPTS

• **Human-centered computing → Empirical studies in HCI**.

KEYWORDS

fairness, data science, computational notebooks, Jupyter Notebook

ACM Reference Format:
Galen Harrison, Kevin Bryson, Ahmad Emmanuel Balla Bamba, Luca Dovichi, Aleksander Herrmann Binion, Arthur Borem, and Blase Ur. 2024. JupyterLab in Retrograde: Contextual Notifications That Highlight Fairness and Bias Issues for Data Scientists. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3613904.3642755

*Also with University of Chicago.

CHI '24, May 11–16, 2024, Honolulu, HI, USA

ACM ISBN 979-8-4007-0330-0/24/05
https://doi.org/10.1145/3613904.3642755

1 INTRODUCTION

For decades, researchers have voiced concerns about the (un)fairness of algorithmic decision systems [18, 20]. As data-driven machine learning (ML) systems have been integrated into daily life and used to make automated decisions, these concerns have been realized across many domains [2, 8]. Various solutions have been proposed, including greater transparency [22, 44], fairness auditing [3, 6, 16, 30], and mathematically quantifying fairness [4, 35].

While many current tools for enhancing fairness [6, 23, 30, 44] facilitate post-hoc auditing of a trained model, relevant decisions occur far before that. Creating a model, even without considering fairness, requires a series of decisions about how to clean data, which data to include/exclude, which model architectures to try, and which one of many possible models investigated to deploy. These tacit data-preprocessing decisions significantly affect the fairness and performance of downstream models [19], yet well-intentioned data scientists[1] may not be aware of how these decisions later impact fairness and bias. Tacit decisions tend not to be documented and often rely on individual data scientists' judgment.

In contrast to existing approaches that help data scientists build more fair models through interventions at either the start of the data-science process (e.g., requirements engineering) or the end (e.g., retrospective auditing), in this paper we focus on helping data scientists become more aware of fairness-related impacts throughout the process of turning source data into a generalized model. Absent interventions, decisions data scientists make early in the data-science process may become ossified, with the data scientist unwilling or unable to reconsider them [1].

To this end, we designed, built, and evaluated Retrograde, an open-source[2] extension to the JupyterLab computational notebook

[1]We use the term "data scientist" to refer broadly to anyone engaged in data analysis, feature engineering, or the production of machine learning models.
[2]Our source code is available at https://github.com/UChicagoSUPERgroup/retrograde

environment for Python. Retrograde assists data scientists by generating and displaying contextually relevant notifications throughout the data science process. These notifications highlight potential fairness impacts of the data scientist's decisions throughout exploratory data analysis, data cleaning, model building, and model selection. Our design aims to highlight fairness decisions as they occur and to stimulate re-examination, hence the name Retrograde.

Beyond designing the user experience of the notifications themselves, creating Retrograde required designing and implementing a new back-end framework. As detailed in Section 3.1, this framework overcomes key technical challenges that would otherwise make it onerous or even impossible to calculate the information Retrograde's notifications display. After reading data into a JupyterLab notebook, data scientists frequently clean and otherwise modify the data. As part of this process, they often drop columns representing data subjects' demographics, preventing later analyses of a model's performance relative to those demographics, especially if future actions (e.g., dropping rows with missing data) further modify a `DataFrame`'s rows or columns. Data scientists also often overwrite a given variable (e.g., `X_test`, `df`) while iteratively training different machine learning models, obscuring the relationship between different versions of data.

Retrograde overcomes these challenges by tracing data provenance and data versioning in JupyterLab via continual code analysis monitoring a data scientist's actions. To display notifications at appropriate times, Retrograde focuses on two widely used Python libraries: `pandas` (`DataFrame` manipulation) and `scikit-learn` (ML). When it detects actions like importing new data or training a new ML classifier, Retrograde decides whether to show a customized, data-driven notification about possible fairness shortcomings. Specifically, we designed notifications (see Section 3.3) that alert data scientists about protected classes (e.g., race, gender) and potential proxy variables in their data, missing data (including correlations with demographics), demographic differences in model performance, and counterfactuals for data subjects.

We evaluated Retrograde through a between-subjects online study (Sections 4–5) in which we tasked 51 data scientists—a difficult population to recruit—to use a dataset we provided to build a classifier that would automatically approve or deny loan applications. We assigned each participant either to see Retrograde's notifications continuously throughout the data science process, only at the end, or not at all (see Section 4.2).

In analyzing participants' code, final models, and survey responses, we found that Retrograde influenced participants' reactions, actions, and perceptions. Compared to those who were not given Retrograde, participants who saw Retrograde's notifications throughout the process were far less likely to use protected classes (e.g., race) as predictive features in their models, as well as more likely to impute missing values rather than drop entire rows with missing data. As a whole, those participants also minimized differences in model performance (F1 scores) across racial groups. For all of these cases, some participants specifically attributed these actions to information they learned from Retrograde's notifications. Notably, participants who were not given Retrograde did not manually compute the key information Retrograde's notifications provided, leaving them unaware of the potential fairness issues flagged. Finally, in an end-of-study survey, Retrograde participants were less comfortable deploying the model they had built and more skeptical of their model than participants without Retrograde.

In sum, our work presents and validates a novel approach to automatically notifying a data scientist about specific fairness issues in their data and models in a JupyterLab computational notebook. The front-end notifications we designed were enabled by our back-end framework tracking data provenance and data versioning. We envision Retrograde being a key resource for helping inexperienced data scientists better consider fairness throughout the data science lifecycle, as well as a valuable tool for automatically surfacing potential fairness problems early in the process of data work.

2 RELATED WORK

We highlight prior work on interventions that flag issues in bias and fairness, as well as prior studies of computational notebooks like JupyterLab, including tools for tracing data provenance.

2.1 Interventions Promoting Fairness

Existing tools to help data scientists make decisions about fairness focus on auditing the final model itself. Specifically, they emphasize understanding and debugging model behavior, mostly overlooking how decisions earlier in the data science process contributed to the model. For example, the Google What-If tool [23] enables visual exploration of model performance. Most directly, it supports counterfactual analysis, enabling the data scientist to investigate what features would need to be different for a data point to be classified (predicted) differently. The IBM AI Fairness 360 library (AIF360) [6] implements a variety of fairness metrics and fairness algorithms proposed in the literature. Building off the popular `scikit-learn` library, which is also a focus of Retrograde, AIF360 provides ready implementations for data scientists seeking to audit machine learning models and attempt to correct statistical biases discovered. Fairkit-learn [30] provides a visual interface on top of AIF360 for exploring different metrics and tradeoffs for models.

These tools are used only with an already trained model and do not connect any of their auditing to decisions made earlier in the data science process. Independent evaluations of these toolkits have underscored these points. Richardson et al. asked data scientists to evaluate existing models using two fairness toolkits [48], while Deng et al. asked participants to build a model from data and explore their model using AIF360 and Fairkit-learn [16]. Both studies concluded that existing toolkits focus nearly exclusively on the model-building component of the data science process, not addressing contextual factors related to fairness. Further studies of Fairkit-learn and AIF360 found they can encourage a "checkbox culture" where data scientists do not seek to understand the underlying reasons for disparities identified [3]. In contrast, Retrograde connects issues in a model to the data scientist's earlier decisions.

A complementary approach to fairness emphasizes procedural checks. While the specific procedures vary [40, 47, 50], these approaches rely on somebody with the power to inspect a model and decide whether it should be deployed. These approaches seek to determine how a model will behave when deployed, especially whether that behavior aligns with the values the overseer aims to uphold. Researchers have proposed short summary information sheets describing relevant aspects of the model [44] or the

data [22, 38] to support such processes. While procedural and organizational aspects are critical to fairness, we focus on individual data scientists. A single data scientist's specific actions and analyses can be audited only via comprehensive and burdensome oversight processes that rarely occur in practice [3, 17].

Some work has sought to understand the role tacit decisions, such as the data preprocessing steps Retrograde considers, play in the data science process [12]. Much of this work has been descriptive in nature, often involving interviews. Through semi-structured interviews with data science workers, Muller et al. [45] found that data science necessarily involves a significant amount of intervention and decision making by data scientists, even up to the creation of ground truth. At the same time, Sambasivan et al. [49] found that an emphasis on models, as opposed to data quality, can lead to cascading issues with deployed models. It is no surprise, then, that studies of existing fairness toolkits have called for new tools with more awareness of the end-to-end data lifecycle [16, 37, 48].

Work has also investigated interfaces and visualizations to help data scientists understand model errors. FairVis [9] aims to enable visual exploration of groups with disparate model predictions, while Silva [54] facilitates causal exploration of datasets. Others have sought to enable more general understanding. For instance, Symphony provides multiple views of computational notebooks to different stakeholders [5], facilitating collaboration.

Other systems have sought to more seamlessly link visualization and code [33, 52, 53]. Cabrera et al. proposed a framework for how data scientists try to understand model behavior and implemented AIFinnity, a tool that supports hypothesis-based exploration [11]. Zeno is a user interface and associated Python API that allows users to conduct different forms of error exploration [10]. Chameleon is a system for understanding model performance under different versions of the data [29], which overlaps in small ways with one notification we designed. These tools focus on understanding model errors and therefore become relevant only after data-processing decisions have already been made.

Furthermore, most existing toolkits require that users read technical documentation or have prior knowledge of fairness concepts. Miceli et al. present qualitative evidence that documentation feels burdensome to data practitioners [43]. In contrast, Retrograde's notifications aim to be self-contained. With Retrograde, we expand the scope of fairness tools by enabling contextual notifications and interventions throughout the data science process.

2.2 Computational Notebooks

We built Retrograde on top of the JupyterLab computational notebook environment. Computational notebooks are often used for data exploration, manipulation, feature engineering, and training by practitioners of all levels [31]. They allow users to execute blocks of code in arbitrary order, as well as to inspect different code outputs. This flexibility makes them well-suited to exploring data sets, yet can impair state management and reproducibility [1, 13, 32, 52].

As mentioned in Section 1 and further detailed in Section 3.1, Retrograde's notifications require tracing data provenance and data versions. JupyterLab's aforementioned flexibility makes it especially difficult to reliably map the contents of data, the relationship between different pieces of data (e.g., which data was derived from which other data), and the relationship between specific data versions and specific machine learning models. Because the order of cell executions determines the notebook state, it is insufficient to simply analyze the contents of the notebook in order to understand how a particular model was produced. Some prior work has focused on enabling users to reproduce data artifacts [27, 56], alerting users whenever the notebook contents have changed in ways that prevent recovering the notebook state [39], or predicting user goals from analysis of notebook logs [55]. However, these approaches do not seek to address our key challenges of data and model provenance.

3 THE RETROGRADE ENVIRONMENT

Retrograde is an open-source extension of the JupyterLab environment we developed for displaying data-driven, interactive notifications that help data scientists consider fairness and bias issues. Here, we first describe Retrograde's back-end analysis approach and then describe the notifications we designed on top of this environment.

3.1 Goals and Challenges

Our high-level goal was for Retrograde to provide data-driven notifications that highlight specific fairness considerations based on the data scientist's specific actions throughout the data science process. This high-level goal encompasses several distinct sub-goals. First, we wanted to trace, to the best possible approximation, the process used to take raw data and turn it into a model. Second, we wanted to use this information to infer when to trigger the appropriate notifications. Lastly, we wanted to ensure that the tracing accurately captured the process by which the model was constructed.

Achieving these goals required us to overcome key challenges. Whenever an assignment statement or function (e.g., `.dropna()`) changed the contents of a variable, we needed to record that variable's state and tag it with a version number. Whenever specific versions of existing variables contributed to another variable (e.g., $\texttt{X\_test}_{v3}$ and $\texttt{y\_test}_{v3}$ were used to train model $\texttt{m}_{v6}$), we needed to record these relationships. We also needed to instrument key functions in the data scientist's workflow. For example, when the `scikit-learn .fit()` function was used to train a model, we needed to record the resultant model to evaluate its performance. Similarly, when the `pandas .read_csv()` function was used to import data, we needed to search it for protected classes and, if found, display the appropriate notification. These connections were paramount. For example, computing canonical fairness metrics for a model requires reliable access to the model itself, evaluation data, and the demographics for each data subject even after the data had been cleaned, split into training and test sets, and potentially had the columns with demographic information removed early in this process. While some prior work has used basic provenance tracking to make computational notebooks more reproducible [27, 39, 56], our approach required substantial additional machinery.

3.2 Retrograde's Back-end Analysis Approach

Figure 1 presents a more concrete example demonstrating these challenges, as well as Retrograde's approach to solving them. In this example, a user loads and cleans some data, uses that data to train a model, modifies the data in a way that overwrites data variable names, and then re-evaluates that data. When the user initially

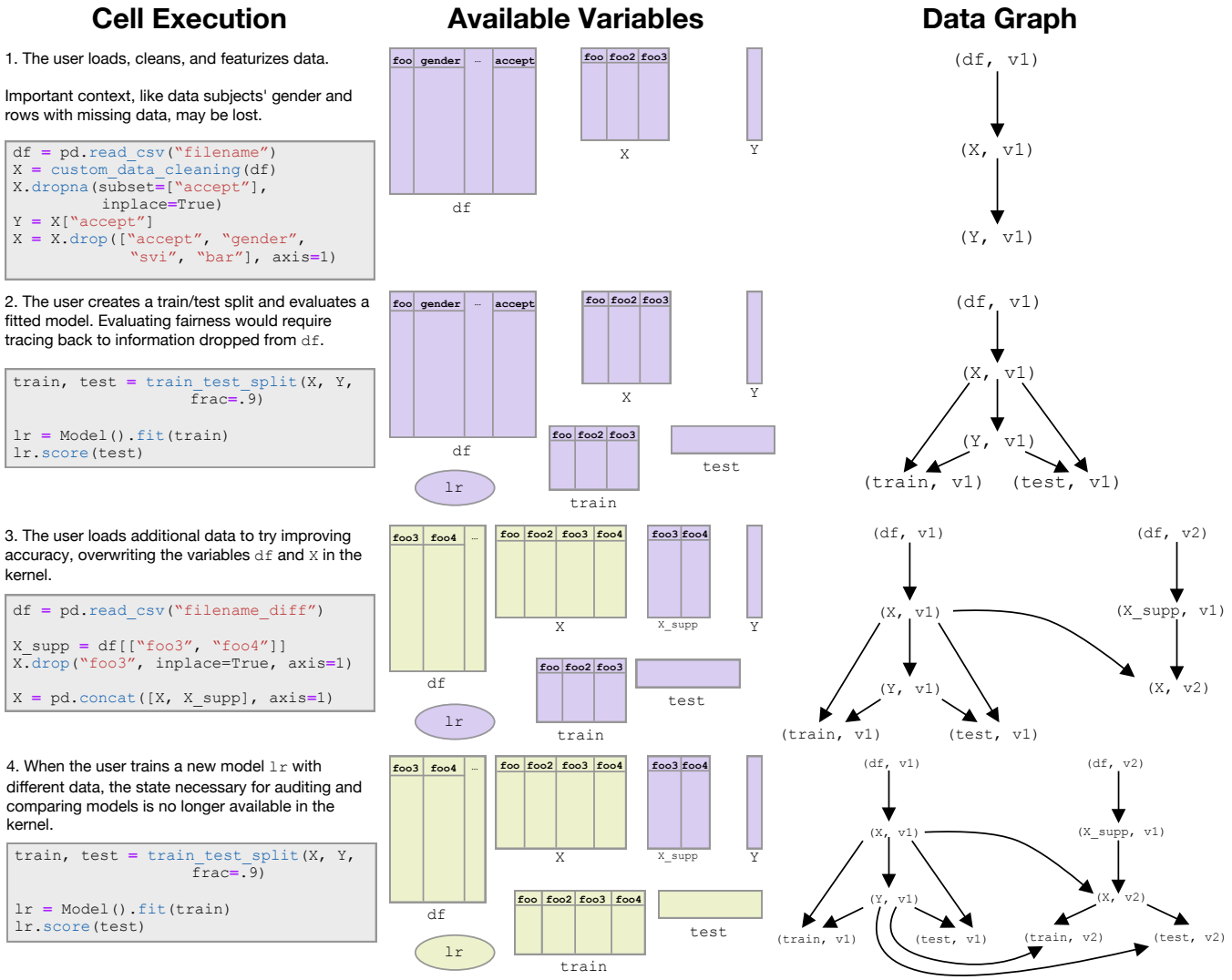


Figure 1: This figure illustrates some key challenges for tracking data provenance and data versions. The left column shows typical example code for importing and cleaning data and then using that data to train a classifier. The center column shows the variable state after the corresponding notebook cell executes. These variables would be available to Retrograde within the kernel at the end of each cell execution. The right column shows a graph abstraction of the data relationships Retrograde captures, distinguishing between versions of each variable. For example, (df, v1) has no direct observable relationship to (df, v2). In the center column, DataFrames colored purple are the first version of the data referred to by their variable name. Those colored green are the second version.

loads DataFrame df, it might contain data that, while inappropriate or irrelevant as a predictor when training a classifier, can provide insight about fairness. For example, if df contains a column encoding a protected class (e.g., gender) that is generally considered impermissible to use as a basis for decision-making, then the data scientist should drop that column when creating a matrix of classification features, often termed X in tutorials. These classification features may undergo further processing and combination, in the example here, by being split into train and test sets. To audit a model's performance on test with respect to the protected class requires mapping rows in test back to rows in df.

Up to Step 2 of Figure 1, this mapping can usually be achieved with just access to the JupyterLab kernel (i.e., the variables defined and available). However, after Step 3, data that could be relevant to the model lr may no longer be defined within the kernel. In other words, df after Step 3 has different values than it did in Steps 1–2.

An approach that does not track the version history of variables cannot fully capture the relevant data relationships. To track these relationships, we developed an extension to the standard iPython kernel used in JupyterLab that makes relevant variables available to Retrograde. We furthermore implemented within Retrograde a data provenance system to capture relationships between specific

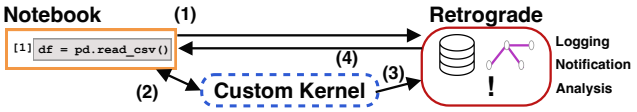


Figure 2: On cell execution, the front-end component (1) sends code to the server extension and (2) kernel. It also saves the *namespace* in a manner accessible to the server extension. The server extension (3) uses the code and namespace to (4) generate notifications.

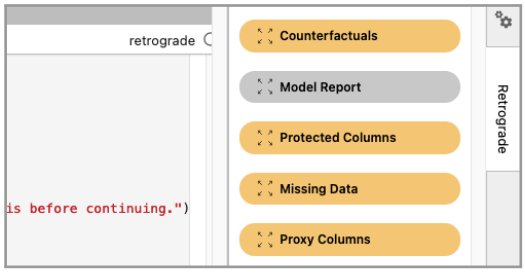


Figure 3: The Retrograde sidebar interface. New notifications are shown in orange and open as a new tab in JupyterLab.

variable versions and specific trained models. The lines of the data graphs on the right side of Figure 1 provide intuition about the relationships our approach captures.

As shown in Figure 2, when a user executes a cell, Retrograde sends the cell contents to the kernel and to our JupyterLab server extension. Retrograde has access to the variables defined within the kernel, as well as the state of the user's Python code at each execution. It uses information about data types and contents derived from the kernel to analyze the code and identify relationships between data. It looks specifically for `Pandas DataFrame` and `Series` objects, but can track transformations of these data back and forth to common data formats like lists and `numpy` arrays.

Retrograde summarizes these data relationships as edges in a data ancestry graph, similar to those in Figure 1's right column. Retrograde also inspects the actual data as defined in the kernel, using this information to determine *data versions*. For example, is `df` in Cell 3 referring to the same data as it was in Cell 2? Retrograde determines this by looking at the data column names, types, and lengths. Retrograde stores information about data versions in a database so that it can refer back to data no longer in the kernel for auditing purposes. This does not fully capture differences and updates to data. For example, rows may be modified without necessarily constituting a new data version. We found that this abstraction, while not fully precise in all cases, enabled the framework to model the data lineage necessary for generating our notifications.

Using this information, Retrograde can issue events, such as when new data is detected or a DataFrame is updated. These events inform notifications, directly or indirectly. Retrograde also tracks `scikit-learn` model fit and model score calls for classifier models, issuing events for these as well. In addition to the tracing and event-listening, Retrograde stores the code executed at each point.

Retrograde focuses entirely on analyzing relationships between `pandas` DataFrame and Series objects and `scikit-learn` models, rather than all of Python. Identifying arbitrary relationships between arbitrary Python variables would be much more computationally expensive, likely impossible in some corner cases, and unnecessary for our goals. We supported `pandas` because it is a very commonly used library for data manipulation and can encode semantically meaningful information about data, such as column names and types. A similar rationale was used to select `scikit-learn`, a machine learning library that requires far less configuration than competing libraries like PyTorch or Keras and is thus popular with beginners. While capable of identifying usage patterns commonly encountered in model development, our analysis is approximate. Retrograde does not enter function calls to inspect data, nor identify global side effects. For example, an assignment `df2 = f(df1)` will always infer a link between `df1` and `df2` regardless of what the function actually does. These conceptual limitations did not appear to impact the information generated during our user study. Furthermore, a computational notebook might contain some cells that are effectively "dead ends," meaning that no further analysis in the notebook builds on those cells' transformations or models. Each fitted model would generate a Retrograde notification. However, after the user moved on to other models, they would not be notified about any new information pertaining to that dead end. Similarly, potentially problematic data manipulations could generate notifications, but Retrograde's provenance tracking would not falsely indicate that a dead-end, problematic version of that data frame was an ancestor of unrelated versions.

3.3 Notifications

Retrograde uses the information derived from the back-end analysis to provide customized, contextual notifications to the data scientist. The figures throughout this section show excerpts of these notifications, while Appendix A provides full screenshots. When a notification is generated, an alert (an orange bubble) appears on the right side of the JupyterLab environment (Figure 3). Alerts are non-blocking. When clicked, they open a new tab presenting the relevant information. Notifications can be triggered either based on actions observed in the notebook or from metadata tags within a notebook. For example, a notification may trigger when a DataFrame with a particular column is first defined in the notebook, when a file is loaded into a dataframe, or when a model is fit. Retrograde also enables metadata tags that identify a particular section of a sample notebook as part of deciding when to show a notification. In our user study, we used a combination of action-based triggers and metadata tags to control notifications. Specifically, we used metadata tags to track a participant's progress through the notebook and provide fine-grained control over when notifications triggered to ensure they appeared at consistent times across participants.

We implemented the following set of notifications, which highlight some fairness-related patterns a data scientist might overlook. They are not designed to provide definitive statements or to otherwise "lint" notebooks for ethics. Instead, these notifications provide a proof-of-concept for the Retrograde platform and were designed to address known challenges associated with common phases of the data science process. We developed the user interaction of these notifications by identifying which tacit decisions are made at each

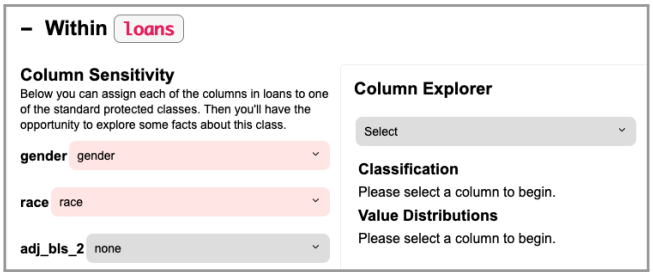


Figure 4: The *Protected Column Note* highlights protected classes that were automatically identified in the DataFrame.

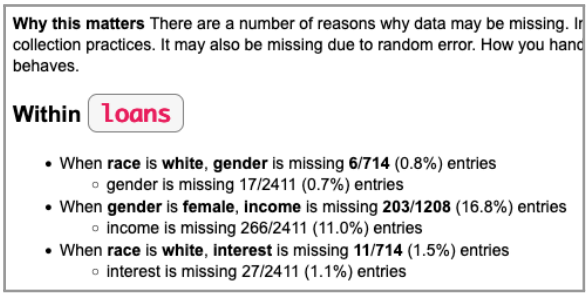
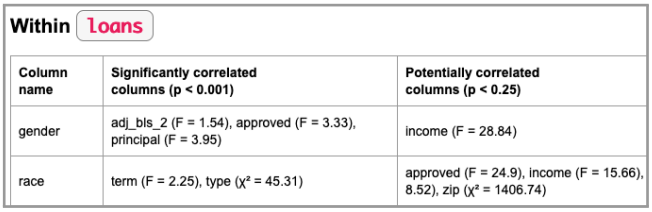


Figure 5: The *Missing Data Note* attempts to find and reveal patterns in missing data related to protected columns.



Within loans

Column name	Significantly correlated columns (p < 0.001)	Potentially correlated columns (p < 0.25)
gender	adj_bls_2 (F = 1.54), approved (F = 3.33), principal (F = 3.95)	income (F = 28.84)
race	term (F = 2.25), type (χ^2 = 45.31)	approved (F = 24.9), income (F = 15.66), 8.52), zip (χ^2 = 1406.74)

Figure 6: The *Proxy Column Note* highlights significant correlations between protected columns and all other columns in the DataFrame.

phase and how they propagate to other phases. In tandem, we iterated on other design decisions through several piloting sessions with experienced data scientists.

3.3.1 Protected Column Note. As shown in Figure 4, the *Protected Column Note* highlights the presence of data that may be impermissible to use as a basis for decision-making. For example, a dataset may include columns for race or gender, which would in certain contexts be illegal to use as predictive features. The *Protected Column Note*'s goal is to notify the data scientist about the presence of such data. As we found in our user study, data scientists commonly use as much data as available without necessarily thinking through the implications of using that data. We wanted data scientists who saw this note to either avoid using demographic data as predictors or to make a contextual determination that the feature did not pose a significant unfairness risk. Other notifications made use of the identified protected classes in auditing (e.g., examining demographic differences in model performance). This notification triggers when a new data or data update event occurs if the data has columns that may represent protected classes. We identify these classes by applying heuristics to the column names (e.g., "gender") and samples of each column's values (e.g., "M," "F," and "NB") as established by US Civil Rights legislation. While using automated heuristics can cause problematic false positives and false negatives, Retrograde asks the data scientist to review and update these automated determinations about protected classes. This note makes use of the data versioning functionality, as well as the kernel inspection functionality. When a new data event or data update event is registered, the *Protected Column Note* applies its heuristics. Furthermore, to address false negatives or incorrect automatic determinations, the interface allows the user to modify the determination for each column, if desired. This note displays the information for all DataFrame objects that could be referenced by the user. That is, if a DataFrame has been defined at any point and has not been deleted from the namespace, then this note contains information on it.

3.3.2 Missing Data Note. As shown in Figure 5, the *Missing Data Note* draws attention to missing data, especially demographic patterns in missing data. If data is missing systematically, simply dropping rows with missing data will introduce bias. Possible mitigations include imputing missing data or being more selective when dropping rows with missing data. The *Missing Data Note* is triggered whenever data is imported or modified if the new or updated DataFrame's missing data is significantly correlated with a protected column, which we defined as having a p-value ≤ 0.2. This approximation captures patterns that may warrant further investigation without overwhelming the user with spurious correlations. Specifically, the *Missing Data Note* first determines whether each column contains categorical or continuous data, which is used to select an appropriate correlation test. This note generates a report for each defined DataFrame with missing data.

3.3.3 Proxy Column Note. We also wanted to alert users about data that is highly correlated with protected classes; these are often termed proxy variables. The *Proxy Column Note* (Figure 6) looks for data that does not explicitly encode a protected class, yet is strongly correlated with a protected class. For example, in the United States, ZIP code is often a strong proxy for race, so using ZIP code as a predictive feature may introduce bias. Rather than trying to detect a set of well-known proxy variables, the *Proxy Column Note* instead empirically calculates the degree to which each variable in a DataFrame correlates with protected classes identified as part of the *Protected Column Note*. Because the appropriateness of using a proxy variable is highly context-dependent [24, 34], we designed the *Proxy Column Note* to raise awareness of identified proxy variables, yet make clear that the data scientist should decide what to do. Even if there were more certainty about the correct mitigation to apply, many mitigations require specialized implementations and analyses beyond the scope of our tool to control for proxy variables. A Retrograde user well-versed in fairness tools might apply some form of fairness correction, such as from the AIF360 library [6].

The *Proxy Column Note* triggers when a dataframe is created (e.g., through a CSV import) or updated. It relies on the *Protected Column Note* for information about which columns represent a protected class. In our user study, we additionally set it to trigger only after the user had finished exploratory data analysis because pilot sessions indicated that this notification triggering at the same time

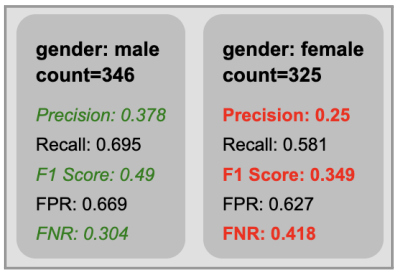


Figure 7: The *Model Report Note* uses data provenance methods to display group-wise error metrics.

prediction	index	income	interest	principal
~~true~~ ➜ false	1981	33432	6.329	~~221671~~ ➜ 117260.681
~~false~~ ➜ true	1258	5279	3.844	~~14213~~ ➜ 118623.319
~~true~~ ➜ false	148	104748	0.415	~~628122~~ ➜ 523711.681
~~false~~ ➜ true	2049	3677	7.962	~~62~~ ➜ 104472.319

Figure 8: The *Counterfactual Note* randomly perturbs up to two features at a time and shows the "prediction diff."

as the *Protected Column Note* overwhelmed participants. In terms of implementation, the *Proxy Column Note* calculates pairwise correlation tests between each non-protected and each protected column. We use heuristics to infer each column's data type (categorical or numeric), using appropriate tests to flag significant correlations. Specifically, we used the nonparametric Spearman's rank correlation coefficient (ρ) to compare numeric data with numeric data, the χ^2 test to compare categorical data with categorical data, and the ANOVA test to compare numerical data with categorical data. This note generates a report for each DataFrame that has been defined in the namespace and that has been identified as having protected classes and non-protected (but potentially correlated) data.

3.3.4 Model Report Note. Depicted in Figure 7, the *Model Report Note* computes fairness metrics with respect to protected columns, as defined by the *Protected Column Note*. Our goal was to nudge participants to analyze their model relative to data subjects' demographics. Like the `metrics` methods of the `scikit-learn` library, our notification computes a model's precision, recall, F1 score, false positive rate, and false negative rate. Whereas standard `metrics` methods compute these values overall, our *Model Report Note* also computes them for each protected class. Crucially, this notification leverages Retrograde's provenance-tracing features to compute these values even if the protected columns were previously discarded from the DataFrame. A data scientist who encounters this note might consider applying a fairness library, reconsider the use of proxy columns identified in the *Proxy Column Note*, or choose a model that minimizes differences across demographics groups.

The *Model Report Note* triggers when a model is evaluated (specifically, when a classifier has the `.score()` method called). Furthermore, it only appears if Retrograde can trace an ancestor DataFrame with at least one protected column to the arguments passed to the score function. In other words, this message is only relevant if Retrograde can make a reasonable guess about which rows in a model's test data corresponded to specific protected classes. This notification automatically updates when the model is re-evaluated with new data. The *Model Report Note* makes use of the columns defined in the *Protected Column Note*, the analysis of the most recently executed code (to identify model `score` calls), and the data version graph. Retrograde traces the arguments to the `score` function backwards until it finds a DataFrame version with protected columns. The note then does best-effort matching between the dataframe version and the test data to get protected class labels for the test set. While this matching is brittle, simple heuristics like index and shared columns worked well in practice.

3.3.5 Counterfactual Note. The idea of a counterfactual explanation is to show how a data subject could have received a different outcome by describing a "similar" data point with a different classification decision [23]. The *Counterfactual Note* (Figure 8) helps the data scientist understand model features that substantially change predictions when perturbed in small ways. A classifier that is highly sensitive to minor perturbations, particularly in apparently irrelevant features, may not be fair [7, 21]. With this notification, we hoped to prompt users to consider the features they had chosen and to select models they expected to be more robust. Similar to the *Model Report Note*, the *Counterfactual Note* triggered on model evaluation. Unlike the *Model Report Note*, it does not rely on protected attributes from the *Protected Column Note* and therefore does not use the data version graph. The *Counterfactual Note* accesses the model and test data, re-evaluating the model after randomly perturbing the evaluation data. In order to make the perturbations of numerical features feasible and minimal for each loan applicant, we decided to randomly add to, or subtract from, each cell up to half the standard deviation of that column. For categorical variables, we randomly select a different value for each row. A user can select columns to modify, seeing which perturbations result in outcomes different from the model's original predictions. A short digest shows the original accuracy, the new accuracy, and the number of predictions changed in each direction.

4 USER STUDY METHODS

We evaluated Retrograde through a two-hour online user study in which participants completed a data cleaning and model-building task using data that deliberately introduced fairness issues. Participants assumed the role of a data scientist tasked to build a model using the provided data to make automated loan determinations. Specifically, we asked them to create a model that would decide whether to grant a loan with "no human oversight or intervention." We asked participants to treat the task with "the same care and rigor as if these tasks were part of [their] job duties." We did not specifically discuss criteria like accuracy or fairness that the model should satisfy. We gave participants a structured Jupyter Notebook (see Figure 9), but did not require them to follow that structure.

4.1 Recruitment, Compensation, and Ethics

We recruited participants from the US and UK on the Prolific crowdsourcing service and Upwork freelancer marketplace. On Prolific, per the platform's policies, we ran a screening survey to identify a

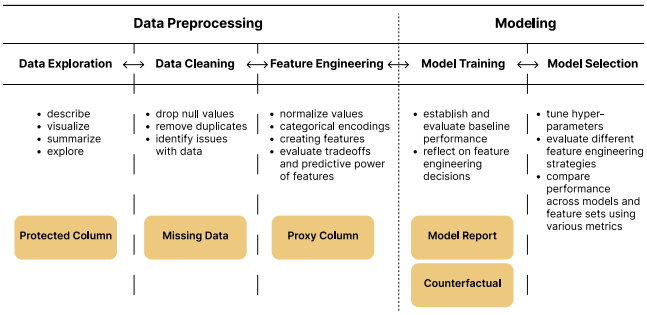


Figure 9: We split the task into five sections. The Retrograde notifications that appeared at each Continuous stage are listed in orange.

Table 1: The columns in the synthetic data we provided and how they were described to participants, who used this data to train a classifier for automating loan decisions.

Name	Description for Participants
ID	An identifier internal to the loan provider
Date	(No description)
Race	The race of the applicant
Gender	The gender identity of the applicant
ZIP	The ZIP code of the applicant's permanent address
Income	The household income of the applicant
Type	The type of loan requested
Interest	The price of taking the loan in percent
Term	Duration of the loan being applied for, in months
Principal	The initial size of the loan in USD
Adj_bls_2	An adjustment factor internal to the loan provider
Approved	Whether the loan provider approved the loan application

pool of participants who reported experience with Python, `pandas`, and `scikit-learn`. We additionally used two screening questions from prior work [15] to assess programming knowledge. We compensated all participants $0.35 for the pre-screen, which typically took well under one minute. Of 4,771 people who completed the pre-screen, only 207 qualified for the study. Of these 207 eligible Prolific workers, 42 completed the study. On Upwork, where data work similar in character to our study is common but screening surveys are not, we did not test for programming knowledge. Instead, we relied on the freelancer's self-described abilities programming in Python and using the relevant libraries. Otherwise, participants on Upwork and Prolific were given identical descriptions of the task. An additional 9 freelancers from Upwork completed the study, resulting in a total of 51 participants. Section 5.1 further describes participants' backgrounds and demographics.

The study took approximately two hours, and we compensated participants $80. We set this amount by examining hourly rates for freelancers with `scikit-learn` experience on sites like Upwork. Our institution's IRB approved our protocol. Our custom JupyterLab instance with the Retrograde environment installed ran on a server we controlled. Participants accessed our server via their web browser; no new software was installed on participants' devices.

4.2 Conditions

Each participant was randomly assigned to one of three conditions. As a baseline, participants in the **None** condition did not receive any Retrograde notifications. The **Continuous** condition presented Retrograde notifications throughout the task when triggered, representing our proposed use case for Retrograde. As an additional control, participants in the **Post-Facto** condition received Retrograde notifications, but only in bulk in the model selection stage near the end of the study. In the Post-Facto condition, participants could still take action in response to the notifications. However, we hypothesized they might be reluctant to return to early data-cleaning and model-building tasks and that other decisions might have otherwise already calcified. In other words, while Continuous best represents our proposed approach to using Retrograde, Post-Facto roughly approximates the temporal behavior of most existing ML fairness tools [6, 44, 51] in which a data scientist evaluates their model only after it has been trained, albeit with additional data. In our study, 17 participants were assigned to each condition.

4.3 Detailed Description of the Task and Data

We provided participants with a dataset, a data dictionary, and a JupyterLab environment with the minimal guidance necessary for completing each section. Rather than using an existing dataset, we artificially constructed the dataset for this study to deliberately introduce fairness-relevant aspects to the data. Real datasets that are commonly used as benchmarks, such as the COMPAS dataset [36], have already been subject to cleaning and aggregation, which are parts of the data science process we wanted to investigate. Our dataset, whose columns are summarized in Table 1, purportedly reflected past creditworthiness decisions. We generated this data to model a scenario where, due to red-lining, there were patterns of geographic- and employment-related discrimination based on data from a large city in the US. The demographic variables probabilistically determined the income, ZIP code, and type of loan being requested for each applicant. We calculated the loan approval column by applying a facially neutral deterministic rule that did not directly rely on any of the demographic attributes. This facially neutral rule, however, created disparate impact: the likelihood of loan approval for Black applicants was 13%, versus 60% for white applicants in our dataset.

In addition to the discriminatory pattern, we also added data cleanliness issues, some of which were also relevant to fairness and bias. We selectively removed data partly at random and partly correlated to gender. We duplicated a small number of rows and mapped certain data to inconsistent values. For example, Black applicants were recorded in the data as "black," "Black," or "African-American." We additionally introduced a data column (*Adj_bls_2*) with an ambiguous name and description. At minimum, the missing data ensured that participants could not simply "click through" without manipulating the data as most `scikit-learn` models do not permit data with null entries.

4.4 Survey Instrument

We asked participants a series of survey questions after each section (see Figure 9), as well as after submitting their final model. Questions ranged from Likert-scale questions to free-response questions. For example, during the Data Cleaning section of the task, we asked participants to "describe the steps you took to clean the data" and answer "what issues and other problematic features did you end up identifying in the data?" Note that we asked participants

only to describe their process; we chose not to prompt them about any fairness- or ethics-related aspects until the end of the study, preferring participants to raise these issues organically. After participants submitted their final model, they answered questions about the effort they spent on different tasks, as well as to perform a self-assessment of their model across different criteria (e.g., the participant's comfort deploying their final model). In this final section, we did specifically ask participants to reflect on the fairness of their model. For participants who saw Retrograde notifications (Continuous, Post-Facto), we asked their impressions of the notifications. Our supplementary materials contain the full survey instrument.

4.5 Data Analysis

To evaluate Retrograde's impact, we holistically analyzed the characteristics of participants' final models, the Python code and comments in their JupyterLab notebooks, and their survey responses.

Upon completing data collection, two members of the research team manually analyzed each participant's notebook to identify which preprocessing and data-cleaning methods the participant used (e.g., dropping all rows with null values), which features the participant used as predictive variables in their models, and which model architectures (e.g., `RandomForestClassifier`) the participant tested during model selection.

We then focused on the final model the participant submitted as their proposed classifier, recording those same characteristics and final model metrics (e.g., precision, recall, F1 score). Additionally, we used an alternate version of the dataset without the introduced data issues to further evaluate the performance of participants' final models. Our goal was to identify tangible differences in the performance of participants' models. Our backend infrastructure also captured full logs of participants' activities, including cell executions and the time of those executions. These logs helped us understand how model selection progressed.

We analyzed qualitative survey responses in multiple waves. Initially, one author open coded participants' free responses to all questions in each section of the survey, focusing on how participants discussed considering particular actions, why they made certain decisions in the end, and how they reacted to Retrograde's notifications (Continuous and Post-Facto only). Another author then used this codebook to code all the data, modifying and adding codes as needed. These two coders then met and resolved disagreements on all responses. The two researchers then revisited the coded survey responses, iterating through each participant and discussing their responses across questions in light of the particular actions they did and did not take in their code. Beyond this qualitative understanding, for which we present both key themes and representative quotes, we also present quantitative survey data (e.g., Likert-scale responses).

Because each condition had only 17 participants, we would have had very low statistical power in attempting to make statistical comparisons across groups. Note that data scientists with the requisite skills to complete the study are both quite difficult and quite expensive to recruit, especially for a two-hour task. As a result, it was not realistic to recruit a larger sample. That only a small fraction of Prolific users who completed our screening survey qualified for the study underscores this challenge. Instead, we built a holistic understanding of Retrograde's impacts in part by examining each participant's collected actions in data cleaning and model building alongside their survey responses. In many cases, participants assigned to a condition in which they saw Retrograde's notifications directly attributed actions they took to Retrograde's notifications in their survey responses, highlighting Retrograde's impact. Note that while we present counts of how many participants (out of 17) in each condition took particular actions, these numbers should not be interpreted as generalizable proportions given the small sample.

5 RESULTS

This section describes participants' processes, results, and perceptions throughout the user study. We observed that participants in Post-Facto and None were remarkably likely to use race or gender as predictive features, whereas participants in Continuous were significantly less likely to do so. Participants in Continuous tended to submit models that had less disparity between racial groups in terms of F1 scores. Furthermore, when discussing the process of data science, participants in Continuous were more likely than those in Post-Facto and None to highlight fairness concerns as the basis for decision-making. We first highlight notification-specific findings, followed by more general results.

5.1 Participants

We had a total of 51 participants (17 per condition), with 42 recruited from Prolific and 9 from Upwork. Among participants, 70.6% identified as male, 23.5% as female, and 5.8% preferred not to say. They were young, with 84.3% between the ages of 18 and 34. They identified as White (58.9%), Asian (19.6%), Black (7.8%), mixed (3.9%), and Hispanic/Latine (2.0%); 7.8% preferred not to say.

Participants were highly educated, with most (94.1%) holding a bachelor's degree or higher. The remaining 5.9% of participants were currently university students studying computer science or machine learning. As would be expected given the screening criteria and skills necessary to complete the study, participants reported a high degree of technical skill. Overall, 96.0% of participants held either a degree or a job in a computer programming-related field. Jobs participants held included data scientist for a Fortune 500 company, post-graduate researcher in artificial intelligence, and careers in business analytics, computational physics, and scientific research involving neuroimaging. Participants without significant technical employment experience all reported taking courses in machine learning, data science, or statistics. Participants who did not report significant experience in data science explained that they gained the requisite skills for this task by working in highly related fields (e.g., data engineering) or using machine learning libraries in hobby projects. Examples of hobby projects from such participants included creating bots to trade cryptocurrency and stocks, competing in data science competitions, and training neural networks to identify skin lesions from patient photographs.

5.1.1 Protected Column Note. As seen in Table 2, 12 participants in the None condition and 11 in the Retrograde Post-Facto condition chose to use protected columns in their final model, versus only 5 participants in the Retrograde Continuous condition. Using these protected columns as predictive variables in granting or denying a loan would typically be seen as discriminatory. Participants in the

Table 2: The number of participants among the 17 in each condition who adopted the specified data-cleaning and modeling approaches. What we believe to be the best (most fair) value for each approach is bolded.

Condition	Used Protected Columns	Used Proxy Column (ZIP)	Used .dropna() Function	Imputed Missing Data Values	Cleaned Categories
None	12	9	15	5	**8**
Continuous	**5**	**8**	**11**	**8**	5
Post-Facto	11	11	13	6	7

Continuous condition frequently pointed to the *Proxy Column Note* as the reason they excluded those variables: "*The information from the notifications made me reconsider using race and gender directly in the model*" (P-27-Continuous). P-34-Continuous explained, "*I did not include gender and race [as predictor variables]. The notifications made me realize this was protected data.*" Participants who did not see the *Protected Column Note* during data cleaning (i.e., Post-Facto and None) were also more likely to discuss race or gender as a useful predictor (8 and 7, respectively) in their survey responses.

This did not mean that those who used race or gender as predictors were unaware of shortcomings for their model's fairness. Participants in all groups discussed the usage of race or gender as problematic in some form in their survey responses (10, 8, and 7 from Continuous, Post-Facto and None, respectively). As Table 2 indicates, this did not always translate to removing the data. For instance, P-32-Continuous kept the protected columns despite knowing they were problematic because they improved accuracy.

In contrast, None and Post-Facto participants tended to start with as much data as possible and defer the examination of those decisions for later. With Post-Facto, Retrograde's interventions did not have the same effect on provisional decision-making as in Continuous. For example, P-36-Post-Facto discussed the use of race and gender as problematic, yet used race in their final model: "*Race seems to be a wildcard,*" they explained. "*A shockingly low amount of black people were [accepted] for a loan. That could be because their average income was lower though.*" When asked how well they understood the model and whether the model was biased, they responded, "*It makes decisions using race & sex so that can't be exactly fair. The biggest indicator of a loan being turned down was race.*" That participant epitomized a behavioral pattern frequent among participants in None and Retrograde Post-Facto: provisional decisions becoming permanent despite awareness of them being potentially problematic. P-41-Post-Facto exemplified this phenomenon rather directly, saying that they did not take the notifications into full account because "*they only popped up at the end.*"

Continuous participants also discussed the use of race or gender as problematic, but their stark behavioral differences suggest that the *Protected Column Note* produces changes in thinking and in behavior seen in Table 2 and as demonstrated by comments like the following: "*I think the protected data aspects will make my solution less effective, as I'll probably avoid incorporating them into the decision making*" (P-20-Continuous). Similarly, P-18-Continuous commented, "*I did not use any of protected variables, I was going to at first (though I did use 'proxy' variables as discussed).*"

5.1.2 Missing Data Note. As discussed in Section 4.3 and our supplementary materials, the synthetic data we provided contained three categories of data issues: missing data, inconsistent categorical labels, and duplicated rows. The most straightforward, albeit naive, way to handle missing data is to drop all rows (data subjects) with any missing data, but this can cause systematic biases if missing data is correlated with demographics [25]. The data we gave participants had a gendered pattern of missing data in which women with approved loans frequently had missing income entries.

Participants in Continuous were less likely to use `.dropna()`—dropping all rows with null values—and more likely to impute missing data, both of which are often preferable. Whereas 11 Continuous participants and 13 Post-Facto participants used the `.dropna()` function to drop all rows containing *any* missing data, 15 None participants did so. While 8 Continuous participants imputed missing data, only 6 Post-Facto and 5 None participants did so.

Participants expressed the value in having missing data automatically flagged for them: "*They helped me see how much missing data there was*" (P-21-Continuous). "*Missing data [notifications] allowed me to fix my code for data cleaning*" (P-29-Continuous). P-30-Continuous commented, "*The notification about missing data was really useful, as that would be annoying to do manually in pandas.*"

Some participants found it particularly valuable that the *Missing Data Note* suggested imputing missing data: "*I imputed missing values differently based on the insights provided in 'Missing Data'*" (P-45-Post-Facto). P-18-Continuous did not feel imputing data was sufficient, yet was not sure what to do instead: "*Sometimes I didn't know what action to take, for instance when you told me how biased the missing data was (more female data missing for instance).*" Overall, the *Missing Data Note* was valuable for participants in surfacing information that requires potentially complicated computations to detect correlations between missing data and protected classes.

5.1.3 Proxy Column Note. The proxy column note highlighted potential proxy variables for protected classes. A data scientist should carefully consider whether it is appropriate to include such proxy variables as predictors in their models. While only 8 Continuous participants included proxy variables in their final model, 11 Post-Facto and 9 None participants did so. In their final survey, Retrograde participants again referred frequently to the notifications as the way they were nudged to consider proxy variables. P-20-Continuous explained, "*I was informed that some of the columns I extracted are highly correlated to some protected classes.*"

While participants expressed awareness of the issues noted by the *Proxy Column Note*, they tended to be less confident in how to address the issues that it brought up. For instance, P-35-Post-Facto wrote, "*I removed term as it can act as a proxy variable to race and gender. I kept principal even though it is a proxy as it seemed like it would be difficult to make a decision regarding approving loans without knowing that.*" Discussing additional issues they observed with the data, P-26-Continuous commented, "*[I]would be worried that the 'zip code' would be used as a proxy for race or something similar and be discriminatory. For [the] real world I would probably*

want to spend a long time looking at things like that to determine if that was the case." Other participants also expressed that they might have been more successful addressing proxy columns if given more time: "*If I had more time it would have been useful to engineer features based on their significance in counterfactuals and proxy columns*" (P-45-Post-Facto).

Even for participants who still used proxy variables, Retrograde raised awareness of the issue. In discussing the fairness of their final model, P-27-Continuous wrote, "*I am not using race and gender directly, but these may be correlated with other features I used such as income and zip code.*" In contrast, only two participants in the None condition mentioned either proxy variables or ZIP codes being correlated with race in any of their survey answers.

5.1.4 Model Report Note. The *Model Report Note* highlighted differences in model performance across protected classes. To look at the effects of the *Model Report Note*, we examined potential differences in the performance of the final models participants chose to submit, as well as how participants reported making those decisions. While participants across all groups had roughly similar model performance overall (Table 3), participants who saw the *Model Report Note* were likely to have less disparity across protected classes in F1 scores and other key metrics (Table 4).

Participants' final models in Continuous and Post-Facto had substantially smaller *ranges* in F1 Scores across racial groups (0.19 and 0.23, respectively) than participants in None (0.33). This suggests that while participants largely had similar error rates on the overall dataset, their models differed substantially when looking at performance across racial groups. Note that Continuous and Post-Facto participants saw the *Model Report Note* at roughly the same point.

While participants across all conditions were likely to discuss accuracy as the key factor for choosing their model, Continuous participants were more likely to discuss fairness concerns as a reason not to deploy the model due to the *Model Report Note*. Continuous participants (8) were more likely to express that their fairness concerns outweighed the utility of deploying the model (5 and 2 from Post-Facto and None, respectively). A number of those participants centered the value of the *Model Report Note*: "*I read the model report and then modified the script because it gave me ideas*" (P-39-Post-Facto). "*Thanks to the orange tab called Model Report, I know that my model has much lower precision for females than males, so I need to work on that*" (P-20-Continuous). Similarly, P-25-Continuous explained, "*The model report showed that one of my models performed poorly for different races so I went back to try and improve the performance across the races…I would not have noticed this as it didn't appear in the sklearn metrics.*" For further discussion of Retrograde's impact on deployment perceptions, see Section 5.3.

This result highlights how participants who saw notifications were able to make alterations that did not appreciably affect overall performance, but decreased disparities between sensitive groups. A well-known result states that for an imperfect classifier, it is impossible to simultaneously equalize false positives and false negatives between two groups [14, 35]. This means that the groups will necessarily have disparities in precision (the ratio of true positives to true plus false positives), or in recall (the ratio of true positives to all positive instances). The F1 score is the harmonic mean of precision and recall, so one could choose many different models, each of which strikes a different balance between precision and recall for each group, while maintaining the same F1 score. Participants in Continuous and Post-Facto had much lower F1 ranges than those in None, but had similar ranges for precision and recall. This suggests that the models developed by participants in Continuous and None may have been more equal in some regards, but may have balanced how to prioritize precision or recall disparities differently. In other words, the advice provided by the notifications translated into material differences in model performance.

5.1.5 Counterfactual Note. The *Counterfactual Note* was designed to further facilitate reasoning about uncertainty, specifically regarding model performance on similar (but unseen) data, as well as understanding which data creates the most variability in model predictions. We hoped to promote analysis of model brittleness, possibly identifying sensitive values that strongly impact model predictions. Some participants reported utilizing it as a deciding factor for their final model: "*[The] counterfactuals [notification] was helpful for choosing [the] best model*" (P-29-Continuous). Others found this notification challenging to grasp: "*They seemed like useful info but were a bit overwhelming. I had to look for other resources to try to understand what was being said about counterfactuals a bit more*" (P-35-Post-Facto). One participant specifically cited time as a primary obstacle to acting on the information from the *Counterfactual Note*: "*If I had more time it would have been useful to engineer features based on their significance in counterfactuals and proxy columns*" (P-29-Post-Facto). This notification, more than others, was intended to present information that was not immediately actionable, yet prompts future consideration (see Section 5.3).

5.2 Analysis Without Retrograde

To gauge whether the Continuous version of Retrograde provided participants novel information or information they would have calculated anyway, we examined all None and Post-Facto notebooks to determine if any of those participants manually performed analyses similar to what Retrograde would have shown. Specifically, we coded the notebooks for missing data analysis, pairwise correlations among columns to detect proxy columns, calculating model metrics by demographic group, and calculating counterfactuals or other types of feature importance. Because Retrograde's calculations prioritize empirical relationships between protected classes and the rest of the data, we distinguish when these methods were focused on protected classes or applied more generally. For example, the missing data notification looks at correlations with data subjects' demographics. We placed less importance on general analyses of missing data. Practically all participants checked for null values in the data, the presence of which would prevent a `scikit-learn` classifier from being trained. The *Missing Data Note* was more specific and rigorous in analyzing demographic biases in missing data.

No participants in the None or Post-Facto conditions calculated any of the key information presented in the *Missing Data Note*, *Model Report Note*, or *Counterfactual Note*. In contrast, the *Proxy Column Note*'s calculations were roughly approximated by eight of the 34 None and Post-Facto participants. Seven participants calculated correlations between all columns of the dataframe or only non-sensitive columns, though not in ways that directly reproduced the *Proxy Column Note*'s insights about proxy variables. P-6-None

Table 3: Key model metrics across conditions. The labels *FPR* and *FNR* refer to "false positive rate" and "false negative rate," respectively. *Larger* numbers are preferable for the F1 score, precision and recall. *Smaller* numbers are preferable for the FPR and FNR. The best value for each metric is bolded.

	Condition	F1 Score	Precision	Recall	FPR	FNR
Mean (σ)	**None**	0.48 *(0.28)*	0.70 *(0.16)*	**0.65 *(0.18)***	0.24 *(0.20)*	**0.47 *(0.24)***
	Continuous	**0.52 *(0.29)***	0.73 *(0.14)*	0.60 *(0.20)*	**0.18 *(0.14)***	**0.47 *(0.22)***
	Post-Facto	0.45 *(0.31)*	**0.74 *(0.15)***	0.62 *(0.21)*	0.20 *(0.17)*	0.50 *(0.25)*
Median	**None**	0.52	0.68	**0.68**	0.19	0.54
	Continuous	**0.57**	0.72	0.57	**0.14**	**0.48**
	Post-Facto	0.50	**0.75**	0.65	0.17	0.50

Table 4: The range of key metrics across demographic groups (i.e., the difference between the maximum and minimum values for any particular demographic group) by condition. *Smaller* differences indicate greater equality across demographic groups.

		Range (Max - Min) Across Racial Groups					Range (Max - Min) Across Gender Groups				
	Condition	F1 Score	Precision	Recall	FPR	FNR	F1 Score	Precision	Recall	FPR	FNR
Mean (σ)	**None**	0.33 *(0.14)*	0.30 *(0.16)*	0.31 *(0.22)*	0.16 *(0.12)*	0.31 *(0.22)*	0.14 *(0.10)*	0.21 *(0.13)*	**0.12 *(0.08)***	0.15 *(0.11)*	**0.12 *(0.08)***
	Continuous	**0.19 *(0.08)***	0.25 *(0.13)*	**0.25 *(0.14)***	0.16 *(0.11)*	**0.25 *(0.14)***	**0.12** (0.08)	**0.14 *(0.09)***	**0.12 *(0.08)***	**0.13 *(0.10)***	**0.12 *(0.08)***
	Post-Facto	0.23 *(0.10)*	**0.24 *(0.12)***	0.29 *(0.13)*	**0.15 *(0.08)***	0.29 *(0.13)*	0.14 *(0.13)*	0.20 *(0.17)*	0.16 *(0.12)*	0.14 *(0.09)*	0.16 *(0.12)*
Median	**None**	0.31	0.34	0.21	**0.14**	0.21	0.11	0.16	**0.10**	0.13	**0.10**
	Continuous	**0.22**	0.28	**0.20**	**0.14**	0.14	0.10	**0.11**	0.11	**0.10**	0.11
	Post-Facto	**0.22**	**0.24**	0.27	**0.14**	**0.13**	**0.08**	0.19	0.11	0.11	0.11

produced a scatter matrix of feature correlations of seven variables that included gender, more directly approximating Retrograde.

For the *Missing Data Note*, none of the 34 None or Post-Facto participants specifically examined patterns in what data was missing. That said, P-13-None summarized all rows in which the income value was missing, which could give insight about demographic biases in missing data if the participant is sufficiently attentive to their manual analysis. In contrast, most participants calculated the number of missing rows, and all participants resolved the empty entries either by imputing missing values or dropping rows with missing data. However, the *Missing Data Note* goes further in calculating the rate of missing data for protected groups.

While the *Model Report Note* provides specific model error statistics for each protected class, highlighting significant differences, no participants reproduced this information. Five participants used confusion matrices or error metrics like F1 score, though on the entire dataset, rather than calculated separately by protected class.

The *Counterfactual Note* implements column perturbation to determine which changes most influence model predictions. No participants implemented a counterfactual analysis, but five of the 34 None or Post-Facto participants examined feature importance.

5.3 Retrograde's Effects on Perceptions

To gauge potential differences in participants' perceptions across conditions, we asked a number of self-report survey questions regarding their models and data. As Figure 10 shows, Retrograde had a substantial impact on participants' perceptions of their models, particularly in engendering a healthy skepticism about their models. Whereas 47% of None participants *disagreed* or *strongly disagreed* that they would feel comfortable deploying their final model, 65% of Continuous and 76% of Post-Facto participants similarly disagreed. In other words, they would feel uncomfortable deploying the model they constructed in the study. Most commonly, participants in the None condition wanted either to achieve higher classification accuracy or to try additional modeling or data-analysis techniques before deployment. Across the two Retrograde conditions, these were participants' second and third most common rationales. In contrast, fairness considerations were Retrograde participants' most common source of discomfort with model deployment. For example, P-23-Continuous found their final model to be "*systematically racist and sexist*," P-22-Continuous felt their model "*should be examined to ensure it's not biased*," and P-27-Continuous wanted "*to consider whether the model is fair to all protected classes [...] I would want to spend more time analyzing it – for example, by looking at feature importance*." Other participants emphasized the need for deeper analysis of their model's fairness, sometimes directly mentioning Retrograde's notifications: "*I did not do nearly enough feature engineering or data exploration in an effort to source out bias in the decision making. What if a good portion of the training data had been made by racist individuals? That bias would have been learned by the model. I would like to answer more questions, especially some things I missed. I did not see in my initial data exploration that a lot of females are missing income data*" (P-38-Post-Facto). Similarly, P-50-Post-Facto wrote, "*I would want to spend more time addressing some possible biases in the model in a real-world deployment (Retrograde pointed out some associations with protected classes)*."

As shown in Figure 10, less than one-quarter of participants in both Continuous and Post-Facto agreed with the statement that their final model was the best that could be achieved given the data, whereas nearly 50% of participants in None thought that this was the best model that could be achieved. In contrast, we observed marked similarity across conditions in participants' agreement that the submitted models were biased, highlighting the pervasiveness of participants' broader sociotechnical concerns about biased models.

Another emergent theme was that of uncertainty surrounding the efficacy of preprocessing decisions and the generalizability of their model. Notably, 7 participants Continuous expressed this uncertainty, while 5 from Post-Facto and 3 from None did so. P-27-Continuous wrote, "*The notifications mostly just presented information. They left it up to me to determine what to do. Again, I was unsure*

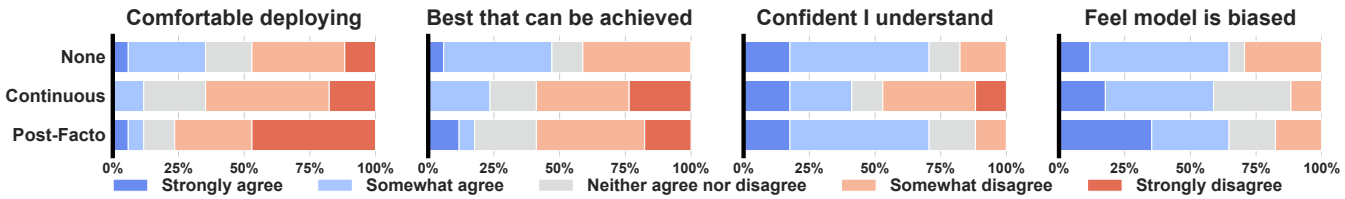


Figure 10: Participants' Likert-scale responses to the following statements: (1) "I would feel comfortable if this model were deployed in the real world for making automated loan decisions"; (2) "I am confident that the model I submitted is the best that can be achieved given the data"; (3) "I am confident that I understand the way the model I submitted makes decisions"; and (4) "I feel that the model I submitted is biased."

what the best approach was to handle protected classes and potentially sensitive information." Similarly, P-18-Continuous commented, "*I don't know how it will behave with outliers or completely different data sets.*" This suggests that Retrograde initiated holistic ethical assessments for some participants and the interventions provided a period to pause and engage with this uncertainty in earnest.

While Retrograde affected some preprocessing decisions, like feature selection—especially for proxy columns—and not dropping all rows with null values, the other data issues were less explicitly affected by the *Missing Data Note* and *Proxy Column Note*. Many participants felt that Retrograde should give more concrete solutions to the topics they covered: "*Some were more clear than others. Some felt a bit overwhelming. We were given the info but not really told how to make decisions based on that info*" (P-35-Post-Facto). Retrograde was designed to be an informational tool, giving few normative recommendations and instead focusing on providing analysis relevant to user-specific notions of sensitivity, which for many in Continuous and Post-Facto (14 and 10, respectively) left them feeling limited or stymied more frequently than those in None (8). Interestingly, for some participants this same sentiment was directed towards other notifications: "*Sometimes, e.g. in the proxy columns or the missing data columns, the actions to take were clear. However in the counterfactuals, protected data and model report, I didn't know what to do with this information*" (P-30-Continuous). These findings suggest that despite Retrograde's interventions uncovering underlying fairness issues and encouraging the data scientist to address them, some data scientists still struggle identifying how to do so.

5.4 Retrograde's Other Effects

Although Retrograde did not provide any explicit feedback about data cleaning beyond missing data, we hypothesized that Retrograde and specifically the *Missing Data Note* might encourage participants to be more conscientious about general data-quality issues. However, we did not observe this to be the case. We examined whether participants fixed cases where multiple values for a categorical valuable arguably ought to be combined, which we term "cleaned categories" in Table 2. In this case, 8 None participants merged these types of categories, whereas only 5 Continuous and 7 Post-Facto participants did so. We observed that the particular choices participants made during data cleaning were highly dependent on the feature-inclusion decisions participants made in the earlier phase of the data analysis. In part this was due to the experimental design; categorical inconsistencies were included only in the race column, which meant that participants who excluded the column early in the process had no reason to notice or address those inconsistencies.

Furthermore, participants' activity logs indicated that many Continuous participants expended the most effort in the initial 20% of the task. In contrast, in Post-Facto, the final 20% of the task saw participants iterating on specific cells the most. In contrast, the None condition saw a relatively even distribution of effort across the task. Taken together, these results suggest that contextual notifications create different focal points for participants, in addition to impacting the decisions made.

6 DISCUSSION

To encourage data scientists to more carefully consider how the models they are building fall short in fairness, we hypothesized that flagging potential issues *throughout* the process of data science would be more effective than the more common post-facto auditing approach embodied by prior work. To that end, we designed and implemented the Retrograde tool, built on a novel data-provenance-tracking back-end framework we developed for JupyterLab. We evaluated Retrograde in a between-subjects experiment in which 51 data scientists developed a binary classifier for loan decisions.

Empirically evaluating tools for our intended user population is difficult. People with the programming and data science abilities required for our study tend to have myriad employment options and thus tend to be harder to recruit than other populations for user research. Nevertheless, our study participants, who were overwhelmingly professionals or active hobbyist coders, do appear to have largely fit the profile of our intended user. Crucially, all participants reported educational background in programming or employment in a related field. Furthermore, most reported significant professional experience in data science or machine learning. All participants are likely to conduct data science in practice. Nonetheless, while these participants had expertise, they were not necessarily industry experts (e.g., those leading ML teams at major companies).

We found that important issues (e.g., systematic biases in missing data, disparate model performance across demographic groups) tended to go unaddressed without prompting. Due to their focus on different portions of the task, participants who saw notifications developed distinct perceptions of their work and the task than those who did not. The majority of Continuous and Post-Facto participants cited Retrograde in discussing the actions they chose. Our holistic and qualitative findings showed that Retrograde's continuous intervention is likely to tangibly refocus data scientists on fairness and bias early in the exploratory process of data science.

A key result of our study is that participants who received notifications during the task were significantly less likely to use columns with protected features than participants who received notifications at the end of the task or no notifications at all. This finding first indicates that the default behavior of participants appeared to be to take as much data as was available and to use it as a predictor without first considering whether or not it was appropriate. Second, it indicates that even participants who were alerted after the fact, who had the opportunity to go back and reconsider this decision, by and large chose not to do so. This pattern lends credence to our hypothesis that raising issues while they are actively being considered will be a more effective prompt than raising them post-hoc.

Another key finding is that participants in Continuous and Post-Facto produced models that on average tended to be more equitable that those that did not see Retrograde. Note that Post-Facto and Continuous participants would have seen the *Model Report Note* at similar times in their process. One possible explanation is that participants who saw the *Model Report Note* were prompted to conduct a more thorough search and ended up selecting a model on some basis other than overall accuracy. While participants largely discussed their models choices in terms of "accuracy," it was not always clear what specific metric they were referencing. In our materials, we were careful to ask participants to produce the best, rather than the most accurate, model. It may be that our participants felt the need to communicate their rationale using a more "objective" framing and accuracy was the term they most frequently used to do so. We lack the data to be able to make a determination, but understanding the discursive strategies data scientists employ to rationalize their decision-making is an avenue for future research.

We also found that participants found the uncertainty in the process difficult to navigate. In particular, the *Missing Data Note* and *Proxy Column Note* were intentionally somewhat ambiguous in directing the data scientist how to mitigate the issues raised. Addressing the issues highlighted by these notifications requires reasoning about causes and mechanisms. For example, in our data, ZIP code was highly correlated with race, but so were the loan term and type. To respond to this information, participants would have needed to theorize about why the pattern may be occurring, subsequently making a determination about whether and how to react. There is a great deal of uncertainty inherent in each of these stages: the theory may be wrong, and the correct response also may be a matter of dispute. In their surveys, participants expressed wanting more specific guidance from the notifications, though in many situations the correct mitigation to bias issues is highly contextual. Trying to give more specific guidance might be counterproductive.

In addition, Retrograde increased many participants' skepticism about their models' suitability for deployment. While some participants who did not feel confident about deploying their models hoped to try other model-building or data-cleaning techniques, often hoping to increase model accuracy, fairness considerations were the most common concern. For some participants, it seemed that additional time for fairness-focused analysis and reflection would have sufficed. Other participants, however, seemed to want to engage in more extensive processes and interventions.

How to address a model's fairness shortcomings is a major open topic for the research community [26]. In some cases, organization-level interventions, deep engagement with different stakeholders, and complex processes might be most appropriate to understand the ethical issues inherent in weighing potentially competing values. While many aspects of fairness-related decisions should not necessarily be left up to individual people, one intervention that might help reduce some aspects of uncertainty would be a more well-developed fairness "toolbox" for data scientists to audit and modify their models. During the time we were developing Retrograde's notifications, we were unable to identify any clear resources that gave succinct, specific guidance we felt would generalize. Specifically, we looked for resources with practical approaches to addressing fairness issues for people who were not necessarily machine learning experts. While there are textbooks focused on fair machine learning [4, 46], these did not provide the kind of guidance that would be useful for our participants. Providing more concrete starting points for addressing fairness issues could help reduce future users' experiences of uncertainty. In any case, we feel that Retrograde has the potential to alert individual data scientists that conversations need to be started in their organization about the fairness implications of the specific models they are building.

One limitation of the current work is that our study evaluated only the actions of individual data scientists. People exist within social power structures that constrain and coerce individual actors; systemic challenges require systemic change [42]. Indeed, as Section 2.1 mentioned, some prior work on fairness processes centers entirely around organizational and procedural methods. In reality, all analyses automated by Retrograde *could* be done manually, though many of these analyses (e.g., subdividing model performance by protected class, generating counterfactuals, performing all pairwise correlations in search of proxy variables) would be onerous and annoying for the data scientist to perform by hand. We contend that even under rigorous oversight regimes, there ultimately will be decisions visible only to individual data scientists. Our goal was to highlight when such decisions may have fairness impacts so that the data scientist can respond appropriately.

Automatically highlighting decisions that impact fairness required several enabling technical contributions. Developing Retrograde required intellectual contributions to computational notebooks in the form of more robust methods for tracing DataFrames' ancestry and reasoning about the relationship between data scattered throughout a computational notebook. In addition to tracing data provenance, it also required developing programmatic understandings of data semantics. The approaches we implemented were tailored to tabular data and classification tasks. Extending these ideas to more general scenarios is a key avenue for future work.

Our Retrograde tool and accompanying user study focused on a specific task and setting: binary classification on tabular data for granting loans. We chose this task and setting because the potential societal impacts are clear and many prior studies on fairness focus on binary classification. However, there are many other possible tasks and settings. Our notifications implicitly centered on notions of *group fairness*, or disparities between different groups of people (often protected classes like gender or race). If a machine learning model were being used to predict a numerical quantity (e.g., using a linear regression model) or one of multiple outcomes rather than only two possible outcomes, a close analogue of our *Model Report Note* would likely be straightforward to construct. For settings other than loans where the primary fairness criterion is to

minimize differences across groups, we similarly expect Retrograde to generalize. Similarly, Retrograde's underlying framework (notifications enabled by provenance tracking and automated analysis of relationships between data and models) would likely remain effective for supporting many different scenarios.

However, notions of fairness beyond group fairness, or even more general notions of data science ethics, might require radically redesigned notifications and completely different data sources. For instance, the current version of Retrograde would not cover concerns about whether the data with which a data scientist is working is demographically representative of a particular population or even accurate in the first place. Similarly, the societal implications of an automated decision system making specific types of incorrect predictions in a given setting, or even the use of automation in that setting in the first place, would fall outside Retrograde's scope. Furthermore, more exploratory use cases could push a tool's goals and design closer to visualization recommendation [41]. In short, the contents, context, timing, and prominence of notifications may need to change across tasks and settings. Regardless, we hope that Retrograde sparks conversations about potential support structures to help data scientists better consider fairness in their workflows.

7 CONCLUSION

We found that Retrograde's notifications throughout the data science process redirected participants' attention to fairness- and bias-oriented aspects of their data and models. This difference in attention resulted in decreased usage of race or gender as direct bases for decision-making, as well as models with some reduced disparities. Retrograde also impacted participants' actions and perceptions in consequential ways. Whereas participants in the None condition often focused on model training, Continuous participants engaged in greater analysis in the preprocessing stage of data work, ultimately leading them to a more critical lens in their perceptions of their models. We also found that Retrograde's notifications increased participants' uncertainty and skepticism about their work.

These results underscore the importance of continuous interventions for inducing meaningful changes in how data scientists approach the preprocessing stages of data science. While it is possible that, working as part of a socio-technical team in an organization with robust institutional safeguards, the data scientist would have eventually addressed the same issues even if fairness was ignored in the early stages, we believe there is strong value in considering fairness and bias throughout the process of data science.

Future work is needed to explore how an interface like Retrograde can interact with a socio-technical team. Problematic decisions early in the process of data science can inadvertently persist when moving from speculative model building to a production environment, and not every organization has robust procedures for auditing models. That said, notifications did have some drawbacks in terms of design and affordances. Specifically, some participants found Retrograde annoying or confusing, while others were disappointed that Retrograde did not provide direct and explicit solutions to the issues it raised. While this finding suggests space for improvements to the notification design, it may also suggest an unavoidable tension in the continuous interaction model between annoying notifications and surfacing important issues in real-time. Throughout the landscape of fairness tools, the focus has largely been on post-facto audits of models, while the preprocessing stage has not received much scrutiny for the critical effect it can have on the resulting fairness of models. Our user study suggests that continuous and contextual interventions can promote healthy skepticism towards model performance, critical ethical reflection, and early identification of ethical dilemmas and remedies.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grants No. CNS-2047827 and No. IIS-1939728.

We would also like to acknowledge that ACM CHI 2024 is being held in Hawai'i and is offering only extremely limited options for remote participation. Nonetheless, we have chosen not to present this work in person due to the impact the conference will likely have on the Kānaka Maoli (Native Hawaiians) and other locals [28]. We encourage future selection processes for conference locations to more carefully consider how our conferences impact local communities.

REFERENCES

[1] Sara Alspaugh, Nava Zokaei, Andrea Liu, Cindy Jin, and Marti A. Hearst. 2019. Futzing and Moseying: Interviews with Professional Data Analysts on Exploration Practices. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019).

[2] Julia Angwin and Jeff Larson. 2016. Machine Bias. *ProPublica.* https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[3] Agathe Balayn, Mireia Yurrita, Jie Yang, and Ujwal Gadiraju. 2023. "Fairness Toolkits, A Checkbox Culture?" On the Factors That Fragment Developer Practices in Handling Algorithmic Harms. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society.*

[4] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2023. *Fairness and Machine Learning: Limitations and Opportunities.* The MIT Press.

[5] Alex Bäuerle, Ángel Alexander Cabrera, Fred Hohman, Megan Maher, David Koski, Xavier Suau, Titus Barik, and Dominik Moritz. 2022. Symphony: Composing Interactive Interfaces for Machine Learning. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems.*

[6] Rachel K.E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, et al. 2018. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias. (2018). https://arxiv.org/abs/1810.01943.

[7] Emily Black, Manish Raghavan, and Solon Barocas. 2022. Model Multiplicity: Opportunities, Concerns, and Solutions. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency.*

[8] Joy Buolamwini and Timnit Gebru. 2018. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency.*

[9] Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. 2019. FAIRVIS: Visual Analytics for Discovering Intersectional Bias in Machine Learning. In *Proceedings of the 2019 IEEE Conference on Visual Analytics Science and Technology.*

[10] Ángel Alexander Cabrera, Erica Fu, Donald Bertucci, Kenneth Holstein, Ameet Talwalkar, Jason I. Hong, and Adam Perer. 2023. Zeno: An Interactive Framework for Behavioral Evaluation of Machine Learning. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems.*

[11] Ángel Alexander Cabrera, Marco Tulio Ribeiro, Bongshin Lee, Robert Deline, Adam Perer, and Steven M. Drucker. 2023. What Did My AI Learn? How Data Scientists Make Sense of Model Behavior. *ACM Transactions on Computer-Human Interaction* 30, 1 (2023).

[12] Inha Cha, Juhyun Oh, Cheul Young Park, Jiyoon Han, and Hwalsuk Lee. 2023. Unlocking the Tacit Knowledge of Data Work in Machine Learning. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems.*

[13] Souti Chattopadhyay, Ishita Prasad, Austin Z. Henley, Anita Sarma, and Titus Barik. 2020. What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems.*

[14] Alexandra Chouldechova. 2017. Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. *Big Data* 5, 2 (2017).

[15] Anastasia Danilova, Alena Naiakshina, Stefan Horstmann, and Matthew Smith. 2021. Do You Really Code? Designing and Evaluating Screening Questions for Online Surveys with Programmers. In *Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering*.

[16] Wesley Hanwen Deng, Manish Nagireddy, Michelle Seng Ah Lee, Jatinder Singh, Zhiwei Steven Wu, Kenneth Holstein, and Haiyi Zhu. 2022. Exploring How Machine Learning Practitioners (Try To) Use Fairness Toolkits. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*.

[17] Wesley Hanwen Deng, Nur Yildirim, Monica Chang, Motahhare Eslami, Kenneth Holstein, and Michael Madaio. 2023. Investigating Practices and Opportunities for Cross-functional Collaboration around AI Fairness in Industry Practice. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*.

[18] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness Through Awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*.

[19] Sorelle A. Friedler, Carlos Scheidegger, Suresh Venkatasubramanian, Sonam Choudhary, Evan P. Hamilton, and Derek Roth. 2019. A Comparative Study of Fairness-Enhancing Interventions in Machine Learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*.

[20] Batya Friedman and Helen Nissenbaum. 1996. Bias in Computer Systems. *ACM Transactions on Information Systems* 14, 3 (1996).

[21] Prakhar Ganesh, Hongyan Chang, Martin Strobel, and Reza Shokri. 2023. On The Impact of Machine Learning Randomness on Group Fairness. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*.

[22] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2021. Datasheets for Datasets. *Commun. ACM* 64, 12 (2021).

[23] Google. 2022. What-If Tool. https://pair-code.github.io/what-if-tool/

[24] Nina Grgić-Hlača, Muhammad Bilal Zafar, Krishna P. Gummadi, and Adrian Weller. 2018. Beyond Distributive Fairness in Algorithmic Decision Making: Feature Selection for Procedurally Fair Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[25] Rolf H. H. Groenwold and Olaf M. Dekkers. 2020. Missing Data: The Impact of What Is Not There. *European Journal of Endocrinology* 183, 4 (2020).

[26] Galen Harrison, Julia Hanson, Christine Jacinto, Julio Ramirez, and Blase Ur. 2020. An Empirical Study on the Perceived Fairness of Realistic, Imperfect Machine Learning Models. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*.

[27] Andrew Head, Fred Hohman, Titus Barik, Steven M. Drucker, and Robert DeLine. 2019. Managing Messes in Computational Notebooks. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*.

[28] Josiah Hester. 2023. Why is CHI in Hawai'i? https://www.chiinhawaii.info/

[29] Fred Hohman, Kanit Wongsuphasawat, Mary Beth Kery, and Kayur Patel. 2020. Understanding and Visualizing Data Iteration in Machine Learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*.

[30] Brittany Johnson, Jesse Bartola, Rico Angell, Katherine Keith, Sam Witty, Stephen J. Giguere, and Yuriy Brun. 2020. Fairkit, Fairkit, on the Wall, Who's the Fairest of Them All? Supporting Data Scientists in Training Fair Models. http://arxiv.org/abs/2012.09951

[31] Kaggle. 2021. 2021 Survey. https://kaggle.com/competitions/kaggle-survey-2021.

[32] Mary Beth Kery, Bonnie E. John, Patrick O'Flaherty, Amber Horvath, and Brad A. Myers. 2019. Towards Effective Foraging by Data Scientists to Find Past Analysis Choices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*.

[33] Mary Beth Kery, Donghao Ren, Fred Hohman, Dominik Moritz, Kanit Wongsuphasawat, and Kayur Patel. 2020. mage: Fluid Moves Between Code and Graphical Work in Computational Notebooks. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*.

[34] Niki Kilbertus, Mateo Rojas Carulla, Giambattista Parascandolo, Moritz Hardt, Dominik Janzing, and Bernhard Schölkopf. 2017. Avoiding Discrimination through Causal Reasoning. In *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems*.

[35] Jon M. Kleinberg, Sendhil Mullainathan, and Manish Raghavan. 2016. Inherent Trade-Offs in the Fair Determination of Risk Scores. http://arxiv.org/abs/1609.05807

[36] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. 2016. How We Analyzed the COMPAS Recidivism Algorithm. *ProPublica*. https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm

[37] Michelle Seng Ah Lee and Jat Singh. 2021. The Landscape and Gaps in Open Source Fairness Toolkits. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*.

[38] Lydia R. Lucchesi, Petra M. Kuhnert, Jenny L. Davis, and Lexing Xie. 2022. Small-set Timelines: A Visual Representation of Data Preprocessing Decisions. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*.

[39] Stephen Macke, Hongpu Gong, Doris Jung-Lin Lee, Andrew Head, Doris Xin, and Aditya Parameswaran. 2021. Fine-grained Lineage for Safer Notebook Interactions. *Proc. VLDB Endow.* 14, 6 (2021).

[40] Michael A. Madaio, Luke Stark, Jennifer Wortman Vaughan, and Hanna Wallach. 2020. Co-Designing Checklists to Understand Organizational Challenges and Opportunities around Fairness in AI. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*.

[41] Andrew McNutt, Gordon Kindlmann, and Michael Correll. 2020. Surfacing Visualization Mirages. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*.

[42] Milagros Miceli, Julian Posada, and Tianling Yang. 2022. Studying Up Machine Learning Data: Why Talk About Bias When We Mean Power? *Proceedings of the ACM on Human-Computer Interaction* 6, GROUP (2022).

[43] Milagros Miceli, Tianling Yang, Laurens Naudts, Martin Schuessler, Diana Serbanescu, and Alex Hanna. 2021. Documenting Computer Vision Datasets: An Invitation to Reflexive Data Practices. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*.

[44] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model Cards for Model Reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*.

[45] Michael Muller, Ingrid Lange, Dakuo Wang, David Piorkowski, Jason Tsay, Q. Vera Liao, Casey Dugan, and Thomas Erickson. 2019. How Data Science Workers Work with Data: Discovery, Capture, Curation, Design, Creation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*.

[46] Aileen Nielsen. 2021. *Practical Fairness: Achieving Fair and Secure Data Models*. O'Reilly.

[47] Inioluwa Deborah Raji, Andrew Smart, Rebecca N. White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. 2020. Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*.

[48] Brianna Richardson, Jean Garcia-Gathright, Samuel F. Way, Jennifer Thom, and Henriette Cramer. 2021. Towards Fairness in Practice: A Practitioner-Oriented Rubric for Evaluating Fair ML Toolkits. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*.

[49] Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M. Aroyo. 2021. "Everyone Wants to Do the Model Work, Not the Data Work": Data Cascades in High-Stakes AI. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*.

[50] Hong Shen, Leijie Wang, Wesley H. Deng, Ciell Brusse, Ronald Velgersdijk, and Haiyi Zhu. 2022. The Model Card Authoring Toolkit: Toward Community-centered, Deliberation-driven AI Design. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*.

[51] Jessie J. Smith, Saleema Amershi, Solon Barocas, Hanna Wallach, and Jennifer Wortman Vaughan. 2022. REAL ML: Recognizing, Exploring, and Articulating Limitations of Machine Learning Research. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*.

[52] April Yi Wang, Will Epperson, Robert A. DeLine, and Steven M. Drucker. 2022. Diff in the Loop: Supporting Data Comparison in Exploratory Data Analysis. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*.

[53] Yifan Wu, Joseph M. Hellerstein, and Arvind Satyanarayan. 2020. B2: Bridging Code and Interactive Visualization in Computational Notebooks. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*.

[54] Jing Nathan Yan, Ziwei Gu, Hubert Lin, and Jeffrey M. Rzeszotarski. 2020. Silva: Interactively Assessing Machine Learning Fairness Using Causality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*.

[55] Jing Nathan Yan, Ziwei Gu, and Jeffrey M. Rzeszotarski. 2021. Tessera: Discretizing Data Analysis Workflows on a Task Level. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*.

[56] Chenyang Yang, Shurui Zhou, Jin L.C. Guo, and Christian Kästner. 2021. Subtle Bugs Everywhere: Generating Documentation for Data Wrangling Code. In *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering*.

A FULL IMAGES OF RETROGRADE'S NOTIFICATIONS

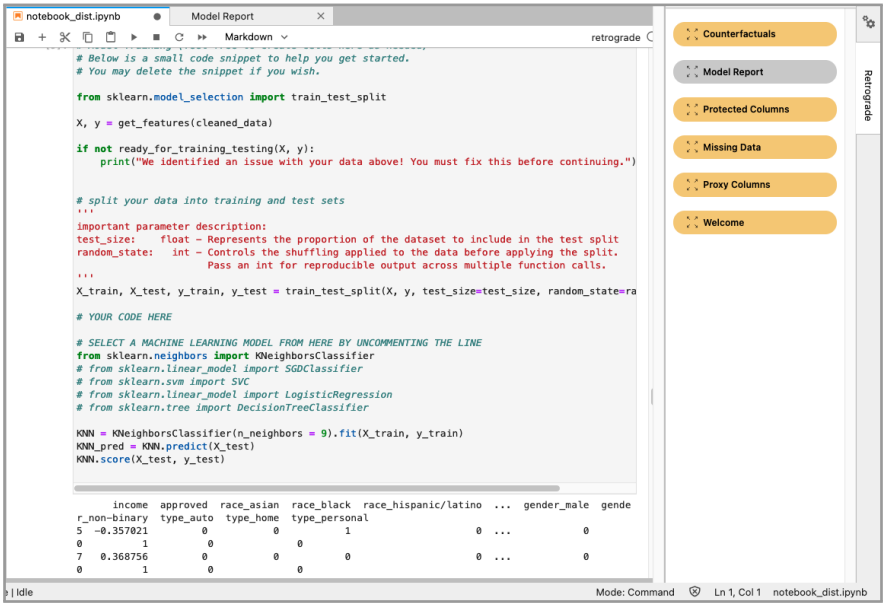


Figure 11: The Retrograde interface. The left side is a standard JupyterLab computational notebook. When a new notification is available, a link appears in orange on the right side, turning gray when it has been read (as the *Model Report Note* was in this example) and orange again if there is an update. The notification window itself appears over the notebook on the left side, persisting as an additional tab (again like the *Model Report Note*) once the user switches back to the notebook tab.

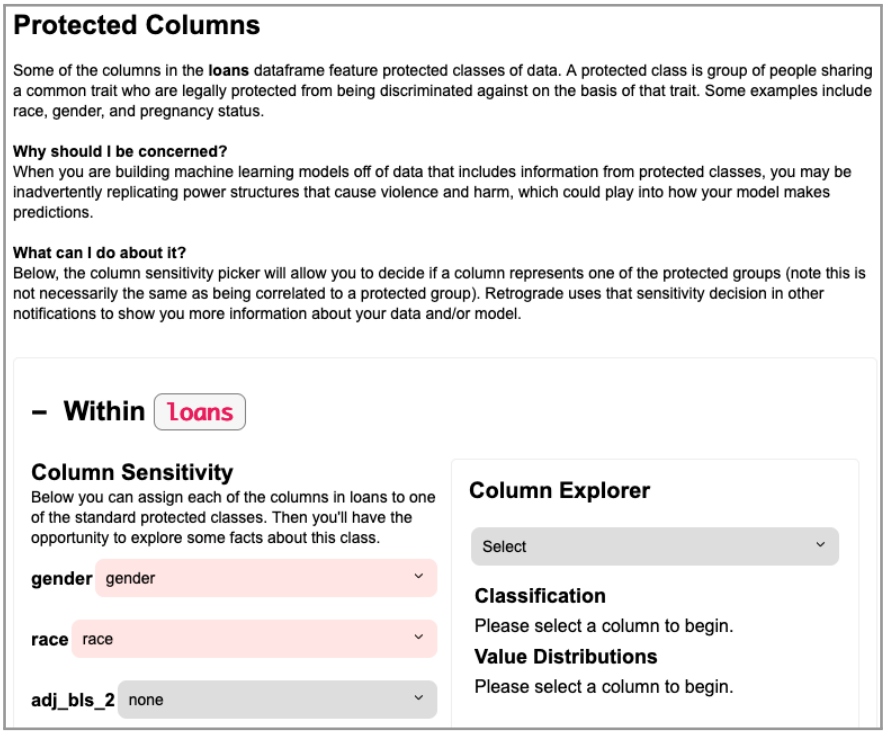
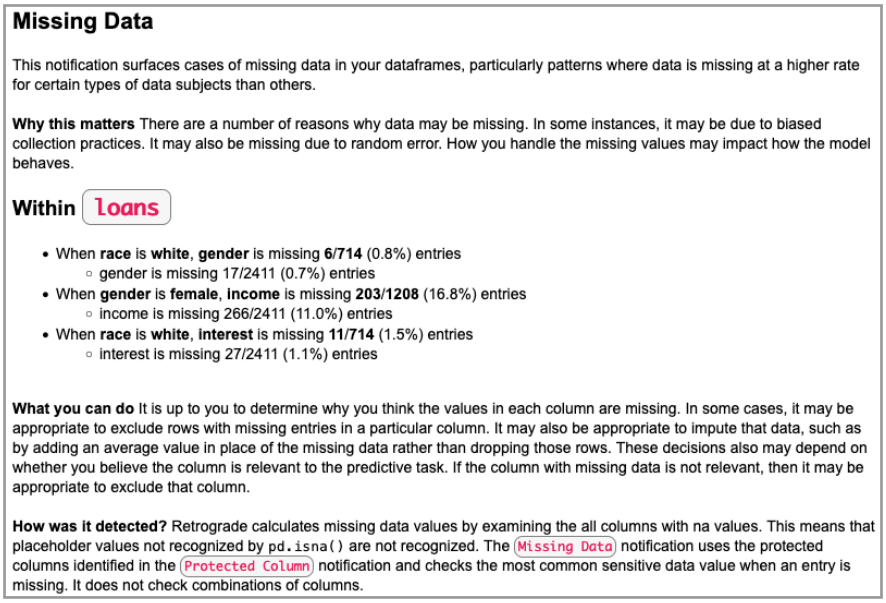


Figure 12: The *Protected Column Note* highlights protected classes that were automatically identified in the DataFrame. The user can inspect the values in each column. Users may manually change the protected category the column falls under.



Missing Data

This notification surfaces cases of missing data in your dataframes, particularly patterns where data is missing at a higher rate for certain types of data subjects than others.

Why this matters There are a number of reasons why data may be missing. In some instances, it may be due to biased collection practices. It may also be missing due to random error. How you handle the missing values may impact how the model behaves.

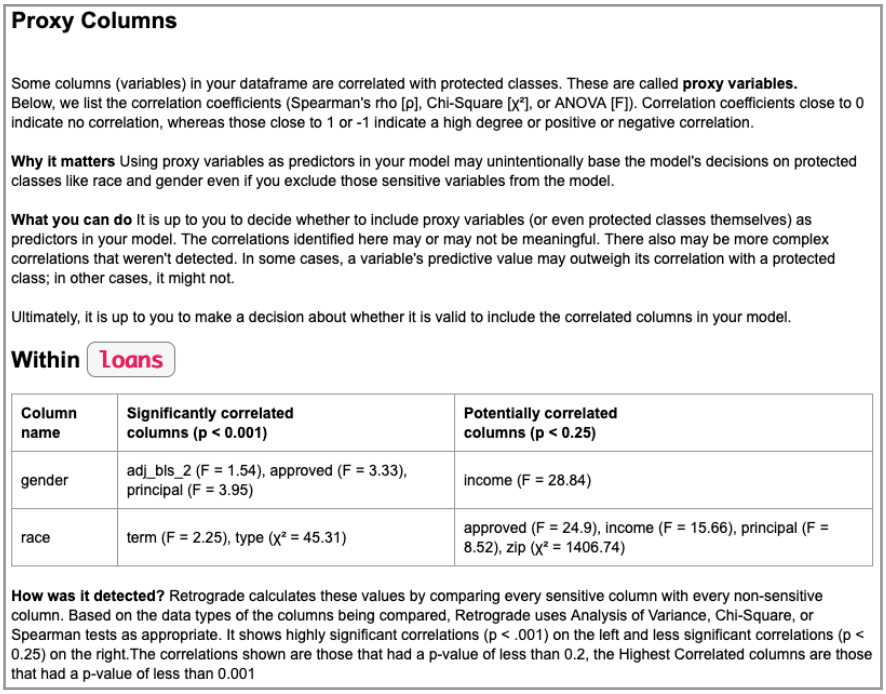
Within `loans`

- When **race** is **white**, **gender** is missing **6/714** (0.8%) entries
  - gender is missing 17/2411 (0.7%) entries
- When **gender** is **female**, **income** is missing **203/1208** (16.8%) entries
  - income is missing 266/2411 (11.0%) entries
- When **race** is **white**, **interest** is missing **11/714** (1.5%) entries
  - interest is missing 27/2411 (1.1%) entries

What you can do It is up to you to determine why you think the values in each column are missing. In some cases, it may be appropriate to exclude rows with missing entries in a particular column. It may also be appropriate to impute that data, such as by adding an average value in place of the missing data rather than dropping those rows. These decisions also may depend on whether you believe the column is relevant to the predictive task. If the column with missing data is not relevant, then it may be appropriate to exclude that column.

How was it detected? Retrograde calculates missing data values by examining the all columns with na values. This means that placeholder values not recognized by `pd.isna()` are not recognized. The `Missing Data` notification uses the protected columns identified in the `Protected Column` notification and checks the most common sensitive data value when an entry is missing. It does not check combinations of columns.

Figure 13: The *Missing Data Note* attempts to reveal patterns in missing data related to protected columns (protected classes).



Proxy Columns

Some columns (variables) in your dataframe are correlated with protected classes. These are called **proxy variables.** Below, we list the correlation coefficients (Spearman's rho [ρ], Chi-Square [χ^2], or ANOVA [F]). Correlation coefficients close to 0 indicate no correlation, whereas those close to 1 or -1 indicate a high degree or positive or negative correlation.

Why it matters Using proxy variables as predictors in your model may unintentionally base the model's decisions on protected classes like race and gender even if you exclude those sensitive variables from the model.

What you can do It is up to you to decide whether to include proxy variables (or even protected classes themselves) as predictors in your model. The correlations identified here may or may not be meaningful. There also may be more complex correlations that weren't detected. In some cases, a variable's predictive value may outweigh its correlation with a protected class; in other cases, it might not.

Ultimately, it is up to you to make a decision about whether it is valid to include the correlated columns in your model.

Within `loans`

Column name	Significantly correlated columns (p < 0.001)	Potentially correlated columns (p < 0.25)
gender	adj_bls_2 (F = 1.54), approved (F = 3.33), principal (F = 3.95)	income (F = 28.84)
race	term (F = 2.25), type (χ^2 = 45.31)	approved (F = 24.9), income (F = 15.66), principal (F = 8.52), zip (χ^2 = 1406.74)

How was it detected? Retrograde calculates these values by comparing every sensitive column with every non-sensitive column. Based on the data types of the columns being compared, Retrograde uses Analysis of Variance, Chi-Square, or Spearman tests as appropriate. It shows highly significant correlations (p < .001) on the left and less significant correlations (p < 0.25) on the right.The correlations shown are those that had a p-value of less than 0.2, the Highest Correlated columns are those that had a p-value of less than 0.001

Figure 14: The *Proxy Column Note* inspects associations between columns identified as protected and other columns in the DataFrame. It tries to apply the most appropriate test and reports correlations with strong, significant correlations.

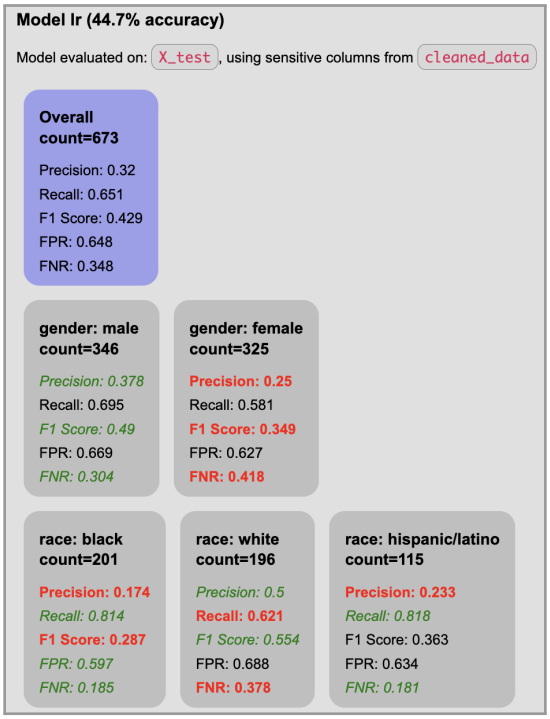
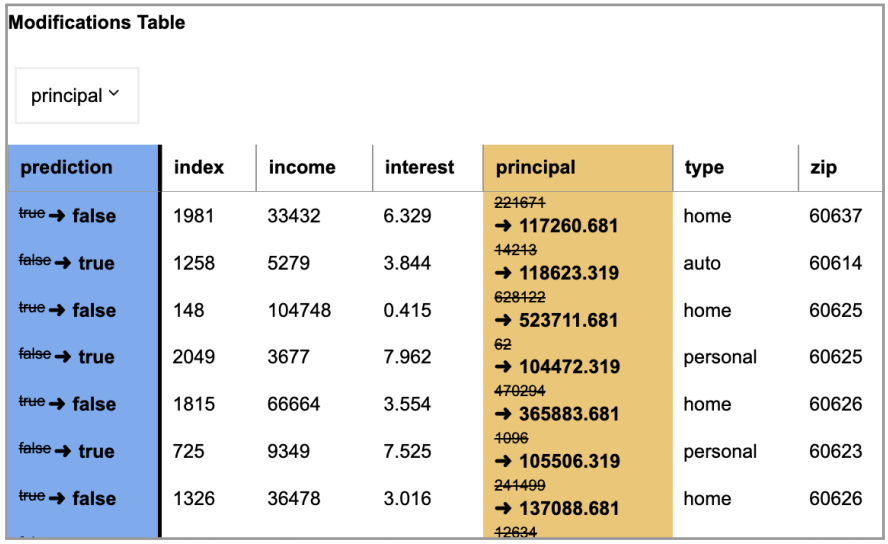


Figure 15: The *Model Report Note* uses our data provenance methods to display group-wise error metrics even when those protected groups' features are not present in the test data (e.g., they were dropped from the DataFrame earlier in the task).



prediction	index	income	interest	principal	type	zip
~~true~~ ➜ false	1981	33432	6.329	~~221671~~ ➜ 117260.681	home	60637
~~false~~ ➜ true	1258	5279	3.844	~~14213~~ ➜ 118623.319	auto	60614
~~true~~ ➜ false	148	104748	0.415	~~628122~~ ➜ 523711.681	home	60625
~~false~~ ➜ true	2049	3677	7.962	~~62~~ ➜ 104472.319	personal	60625
~~true~~ ➜ false	1815	66664	3.554	~~470294~~ ➜ 365883.681	home	60626
~~false~~ ➜ true	725	9349	7.525	~~1096~~ ➜ 105506.319	personal	60623
~~true~~ ➜ false	1326	36478	3.016	~~241499~~ ➜ 137088.681	home	60626

Figure 16: The *Counterfactual Note* randomly perturbs up to two features at a time and shows the "prediction diff" as a table.